## **Understanding the problem without Maven**

There are many problems that we face during the project development. They are discussed below:

**1) Adding set of Jars in each project:** In case of struts, spring, hibernate frameworks, we need to add set of jar files in each project. It must include all the dependencies of jars also.

**2) Creating the right project structure:** We must create the right project structure in servlet, struts etc, otherwise it will not be executed.

**3) Building and Deploying the project:** We must have to build and deploy the project so that it may work.

## **What it does?**

Maven simplifies the above mentioned problems. It does mainly following tasks.

1. It makes a project easy to build
2. It provides uniform build process (maven project can be shared by all the maven projects)
3. It provides project information (log document, cross referenced sources, mailing list, dependency list, unit test reports etc.)
4. It is easy to migrate for new features of Maven

Apache Maven helps to manage

* Builds
* Documentation
* Reporing
* SCMs
* Releases
* Distribution

## **What is Build Tool**

A build tool takes care of everything for building a process. It does following:

* Generates source code (if auto-generated code is used)
* Generates documentation from source code
* Compiles source code
* Packages compiled code into JAR of ZIP file
* Installs the packaged code in local repository, server repository, or central repository

# **Difference between Ant and Maven**

**Ant** and **Maven** both are build tools provided by Apache. The main purpose of these technologies is to ease the build process of a project.

There are many differences between ant and maven that are given below:

|  |  |
| --- | --- |
| **Ant** | **Maven** |
| Ant **doesn't has formal conventions**, so we need to provide information of the project structure in build.xml file. | Maven **has a convention** to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file. |
| Ant is **procedural**, you need to provide information about what to do and when to do through code. You need to provide order. | Maven is **declarative**, everything you define in the pom.xml file. |
| There is **no life cycle** in Ant. | There is **life cycle** in Maven. |
| It is **a tool** box. | It is **a framework**. |
| It is **mainly a build tool**. | It is **mainly a project management tool**. |
| The ant scripts are **not reusable**. | The maven plugins are **reusable**. |
| It is **less preferred** than Maven. | It is **more preferred** than Ant. |

# **How to install Maven on windows**

You can download and install maven on windows, linux and MAC OS platforms. Here, we are going to learn how to install maven on windows OS.

To install maven on windows, you need to perform following steps:

1. Download maven and extract it
2. Add JAVA\_HOME and MAVEN\_HOME in environment variable
3. Add maven path in environment variable
4. Verify Maven

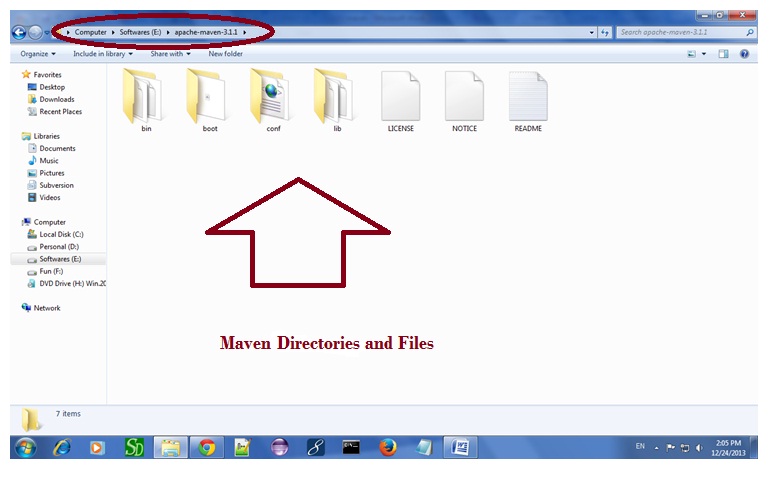
## **1) Download Maven**

To install maven on windows, you need to download apache maven first.

Download Maven latest Maven software from [Download latest version of Maven](http://maven.apache.org/download.cgi)

For example: **apache-maven-3.1.1-bin.zip**

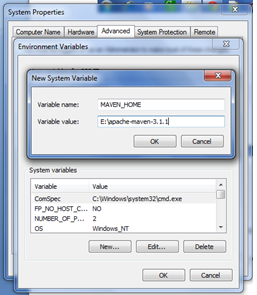
Extract it. Now it will look like this:



## **2) Add MAVEN\_HOME in environment variable**

Right click on **MyComputer** -> **properties** -> **Advanced System Settings** -> **Environment variables** -> **click new button**

Now **add MAVEN\_HOME** in variable name and path of maven in variable value. It must be the home directory of maven i.e. outer directory of bin. For example: **E:\apache-maven-3.1.1** .It is displayed below:



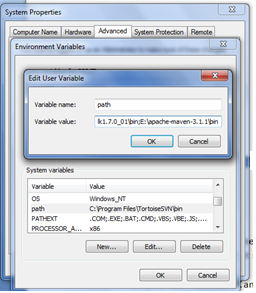
Now click on **OK** button.

## **3) Add Maven Path in environment variable**

Click on new tab if path is not set, then set the path of maven. If it is set, edit the path and append the path of maven.

Here, we have installed JDK and its path is set by default, so we are going to append the path of maven.

The path of maven should be **%maven home%/bin**. For example, **E:\apache-maven-3.1.1\bin** .



## **4)Verify maven**

To verify whether maven is installed or not, open the command prompt and write:

1. mvn −version

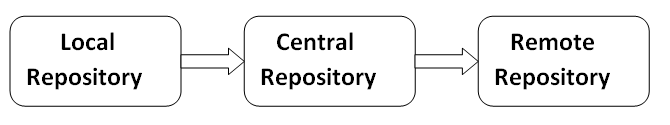
# **Maven Repository**

A **maven repository** is a directory of packaged JAR file with pom.xml file. Maven searches for dependencies in the repositories. There are 3 types of maven repository:

1. Local Repository
2. Central Repository
3. Remote Repository

Maven searches for the dependencies in the following order:

**Local repository** then **Central repository** then **Remote repository**.



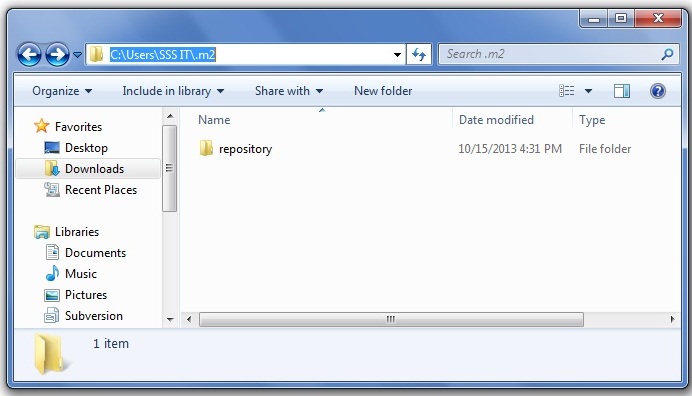
If dependency is not found in these repositories, maven stops processing and throws an error.

## **1) Maven Local Repository**

Maven **local repository** is located in your local system. It is created by the maven when you run any maven command.

By default, maven local repository is %USER\_HOME%/.m2 directory. For example: **C:\Users\SSS IT\.m2**.

By default, maven local repository is %USER\_HOME%/.m2 directory. For example: **C:\Users\SSS IT\.m2**.



## **Update location of Local Repository**

We can change the location of maven local repository by changing the **settings.xml** file. It is located in **MAVEN\_HOME/conf/settings.xml**, for example: **E:\apache-maven-3.1.1\conf\settings.xml**.

Let's see the default code of settings.xml file.

*settings.xml*

1. ...
2. **<settings** xmlns="http://maven.apache.org/SETTINGS/1.0.0"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd"**>**
5. <!-- localRepository
6. | The path to the local repository maven will use to store artifacts.
7. |
8. | Default: ${user.home}/.m2/repository
9. **<localRepository>**/path/to/local/repo**</localRepository>**
10. --**>**
12. ...
13. **</settings>**

Now change the path to local repository. After changing the path of local repository, it will look like this:

*settings.xml*

1. ...
2. **<settings** xmlns="http://maven.apache.org/SETTINGS/1.0.0"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd"**>**
5. **<localRepository>**e:/mavenlocalrepository**</localRepository>**
7. ...
8. **</settings>**

As you can see, now the path of local repository is e:/mavenlocalrepository.

## **2) Maven Central Repository**

Maven **central repository** is located on the web. It has been created by the apache maven community itself.

The path of central repository is: <http://repo1.maven.org/maven2/>.

The central repository contains a lot of common libraries that can be viewed by this url <http://search.maven.org/#browse>.

## **3) Maven Remote Repository**

Maven **remote repository** is located on the web. Most of libraries can be missing from the central repository such as JBoss library etc, so we need to define remote repository in pom.xml file.

Let's see the code to add the jUnit library in pom.xml file.

*pom.xml*

1. **<project** xmlns="http://maven.apache.org/POM/4.0.0"
2. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4. http://maven.apache.org/xsd/maven-4.0.0.xsd"**>**
6. **<modelVersion>**4.0.0**</modelVersion>**
8. **<groupId>**com.javatpoint.application1**</groupId>**
9. **<artifactId>**my-application1**</artifactId>**
10. **<version>**1.0**</version>**
11. **<packaging>**jar**</packaging>**
13. **<name>**Maven Quick Start Archetype**</name>**
14. **<url>**http://maven.apache.org**</url>**
16. **<dependencies>**
17. **<dependency>**
18. **<groupId>**junit**</groupId>**
19. **<artifactId>**junit**</artifactId>**
20. **<version>**4.8.2**</version>**
21. **<scope>**test**</scope>**
22. **</dependency>**
23. **</dependencies>**
25. **</project>**

You can search any repository from Maven official website **mvnrepository.com**.

# **Maven pom.xml file**

**POM** is an acronym for **Project Object Model**. The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.

Maven reads the pom.xml file, then executes the goal.

Before maven 2, it was named as project.xml file. But, since maven 2 (also in maven 3), it is renamed as pom.xml.

## **Elements of maven pom.xml file**

For creating the simple pom.xml file, you need to have following elements:

|  |  |
| --- | --- |
| **Element** | **Description** |
| **project** | It is the root element of pom.xml file. |
| **modelVersion** | It is the sub element of project. It specifies the modelVersion. It should be set to 4.0.0. |
| **groupId** | It is the sub element of project. It specifies the id for the project group. |
| **artifactId** | It is the sub element of project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs. |
| **version** | It is the sub element of project. It specifies the version of the artifact under given group. |

*File: pom.xml*

1. <project xmlns="http://maven.apache.org/POM/4.0.0"
2. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4. http://maven.apache.org/xsd/maven-4.0.0.xsd">
6. <modelVersion>4.0.0</modelVersion>
7. <groupId>com.javatpoint.application1</groupId>
8. <artifactId>my-app</artifactId>
9. <version>1</version>
11. </project>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.javatpoint.application1</groupId>

<artifactId>my-app</artifactId>

<version>1</version>

</project>

## **Maven pom.xml file with additional elements**

Here, we are going to add other elements in pom.xml file such as:

|  |  |
| --- | --- |
| **Element** | **Description** |
| **packaging** | defines packaging type such as jar, war etc. |
| **name** | defines name of the maven project. |
| **url** | defines url of the project. |
| **dependencies** | defines dependencies for this project. |
| **dependency** | defines a dependency. It is used inside dependencies. |
| **scope** | defines scope for this maven project. It can be compile, provided, runtime, test and system. |

*File: pom.xml*

1. <project xmlns="http://maven.apache.org/POM/4.0.0"
2. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4. http://maven.apache.org/xsd/maven-4.0.0.xsd">
6. <modelVersion>4.0.0</modelVersion>
8. <groupId>com.javatpoint.application1</groupId>
9. <artifactId>my-application1</artifactId>
10. <version>1.0</version>
11. <packaging>jar</packaging>
13. <name>Maven Quick Start Archetype</name>
14. <url>http://maven.apache.org</url>
16. <dependencies>
17. <dependency>
18. <groupId>junit</groupId>
19. <artifactId>junit</artifactId>
20. <version>4.8.2</version>
21. <scope>test</scope>
22. </dependency>
23. </dependencies>

</project>

# **Maven Example**

We can create a simple maven example by executing the **archetype:generate** command of **mvn tool**.

To create a simple java project using maven, you need to open command prompt and run the **archetype:generate** command of mvn tool.

#### Syntax

The **syntax** to generate the project architecture is given below:

1. mvn archetype:generate -DgroupId=groupid -DartifactId=artifactid
2. -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=booleanValue

mvn archetype:generate -DgroupId=groupid -DartifactId=artifactid

-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=booleanValue

#### Example

The **example** to generate the project architecture is given below:

1. mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGenerator
2. -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGenerator

-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

#### Note: Here, we are using maven-archetype-quickstart to create simple maven core project. if you use maven-archetype-webapp, it will generate a simple maven web application.

#### Output

Now it will **generate following code in the command prompt**:

#### mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=Cub

#### eGenerator -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=fa

#### lse

#### [INFO] Scanning for projects...

#### [INFO]

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Building Maven Stub Project (No POM) 1

#### [INFO] ------------------------------------------------------------------------

#### [INFO]

#### [INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>

#### >

#### [INFO]

#### [INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<

#### <

#### [INFO]

#### [INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom --

#### -

#### [INFO] Generating project in Batch mode

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar (5 KB at 3.5 KB/se

#### c)

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom (703 B at 0.9 KB/s

#### ec)

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Using following parameters for creating project from Old (1.x) Archetype:

#### maven-archetype-quickstart:1.0

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Parameter: groupId, Value: com.javatpoint

#### [INFO] Parameter: packageName, Value: com.javatpoint

#### [INFO] Parameter: package, Value: com.javatpoint

#### [INFO] Parameter: artifactId, Value: CubeGenerator

#### [INFO] Parameter: basedir, Value: C:\Users\SSS IT

#### [INFO] Parameter: version, Value: 1.0-SNAPSHOT

#### [INFO] project created from Old (1.x) Archetype in dir: C:\Users\SSS IT\CubeGene

#### rator

#### [INFO] ------------------------------------------------------------------------

#### [INFO] BUILD SUCCESS

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Total time: 10.913s

#### [INFO] Finished at: Thu Dec 26 16:45:18 IST 2013

#### [INFO] Final Memory: 9M/25M

#### [INFO] ------------------------------------------------------------------------

#### 'cmd' is not recognized as an internal or external command,

#### operable program or batch file.

#### mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=Cub

#### eGenerator -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=fa

#### lse

#### [INFO] Scanning for projects...

#### [INFO]

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Building Maven Stub Project (No POM) 1

#### [INFO] ------------------------------------------------------------------------

#### [INFO]

#### [INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>

#### >

#### [INFO]

#### [INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<

#### <

#### [INFO]

#### [INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom --

#### -

#### [INFO] Generating project in Batch mode

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar (5 KB at 3.5 KB/se

#### c)

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom (703 B at 0.9 KB/s

#### ec)

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Using following parameters for creating project from Old (1.x) Archetype:

#### maven-archetype-quickstart:1.0

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Parameter: groupId, Value: com.javatpoint

#### [INFO] Parameter: packageName, Value: com.javatpoint

#### [INFO] Parameter: package, Value: com.javatpoint

#### [INFO] Parameter: artifactId, Value: CubeGenerator

#### [INFO] Parameter: basedir, Value: C:\Users\SSS IT

#### [INFO] Parameter: version, Value: 1.0-SNAPSHOT

#### [INFO] project created from Old (1.x) Archetype in dir: C:\Users\SSS IT\CubeGene

#### rator

#### [INFO] ------------------------------------------------------------------------

#### [INFO] BUILD SUCCESS

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Total time: 10.913s

#### [INFO] Finished at: Thu Dec 26 16:45:18 IST 2013

#### [INFO] Final Memory: 9M/25M

#### [INFO] ------------------------------------------------------------------------

#### 'cmd' is not recognized as an internal or external command,

#### operable program or batch file.

#### Generated Directory Structure

Now go to the current directory from where you have executed the mvn command. For example: **C:\Users\SSS IT\CubeGenerator**. You will see that a simple java project is created that has the following directory:

#### CubeGenerator

#### -src

#### --main

#### ---java

#### ----com

#### -----javatpoint

#### ------App.java

#### --test

#### ---java

#### ----com

#### -----javatpoint

#### ------AppTest.java

#### -pom.xml

#### CubeGenerator

#### -src

#### --main

#### ---java

#### ----com

#### -----javatpoint

#### ------App.java

#### --test

#### ---java

#### ----com

#### -----javatpoint

#### ------AppTest.java

#### -pom.xml

As you can see, there are created 3 files pom.xml, App.java and AppTest.java. Let's have a quick look at these files:

#### 1) Automatically Generated pom.xml file

#### <project xmlns="http://maven.apache.org/POM/4.0.0"

#### xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

#### xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

#### http://maven.apache.org/maven-v4\_0\_0.xsd">

#### 

#### <modelVersion>4.0.0</modelVersion>

#### <groupId>com.javatpoint</groupId>

#### <artifactId>CubeGenerator</artifactId>

#### <packaging>jar</packaging>

#### <version>1.0-SNAPSHOT</version>

#### <name>CubeGenerator</name>

#### <url>http://maven.apache.org</url>

#### <dependencies>

#### <dependency>

#### <groupId>junit</groupId>

#### <artifactId>junit</artifactId>

#### <version>3.8.1</version>

#### <scope>test</scope>

#### </dependency>

#### </dependencies>

#### </project>

#### <project xmlns="http://maven.apache.org/POM/4.0.0"

#### xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

#### xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

#### http://maven.apache.org/maven-v4\_0\_0.xsd">

#### <modelVersion>4.0.0</modelVersion>

#### <groupId>com.javatpoint</groupId>

#### <artifactId>CubeGenerator</artifactId>

#### <packaging>jar</packaging>

#### <version>1.0-SNAPSHOT</version>

#### <name>CubeGenerator</name>

#### <url>http://maven.apache.org</url>

#### <dependencies>

#### <dependency>

#### <groupId>junit</groupId>

#### <artifactId>junit</artifactId>

#### <version>3.8.1</version>

#### <scope>test</scope>

#### </dependency>

#### </dependencies>

#### </project>

#### 2) Automatically Generated App.java file

#### package com.javatpoint;

#### /\*\*

#### \* Hello world!

#### \*

#### \*/

#### public class App

#### {

#### public static void main( String[] args )

#### {

#### System.out.println( "Hello World!" );

#### }

#### }

#### package com.javatpoint;

#### /\*\*

#### \* Hello world!

#### \*

#### \*/

#### public class App

#### {

#### public static void main( String[] args )

#### {

#### System.out.println( "Hello World!" );

#### }

#### }

#### 3) Automatically Generated AppTest.java file

#### package com.javatpoint;

#### 

#### import junit.framework.Test;

#### import junit.framework.TestCase;

#### import junit.framework.TestSuite;

#### /\*\*

#### \* Unit test for simple App.

#### \*/

#### public class AppTest

#### extends TestCase

#### {

#### /\*\*

#### \* Create the test case

#### \*

#### \* @param testName name of the test case

#### \*/

#### public AppTest( String testName )

#### {

#### super( testName );

#### }

#### /\*\*

#### \* @return the suite of tests being tested

#### \*/

#### public static Test suite()

#### {

#### return new TestSuite( AppTest.class );

#### }

#### /\*\*

#### \* Rigourous Test :-)

#### \*/

#### public void testApp()

#### {

#### assertTrue( true );

#### }

#### }

#### package com.javatpoint;

#### import junit.framework.Test;

#### import junit.framework.TestCase;

#### import junit.framework.TestSuite;

#### /\*\*

#### \* Unit test for simple App.

#### \*/

#### public class AppTest

#### extends TestCase

#### {

#### /\*\*

#### \* Create the test case

#### \*

#### \* @param testName name of the test case

#### \*/

#### public AppTest( String testName )

#### {

#### super( testName );

#### }

#### /\*\*

#### \* @return the suite of tests being tested

#### \*/

#### public static Test suite()

#### {

#### return new TestSuite( AppTest.class );

#### }

#### /\*\*

#### \* Rigourous Test :-)

#### \*/

#### public void testApp()

#### {

#### assertTrue( true );

#### }

#### }

**Compile the Maven Java Project**

To compile the project, go to the project directory,

for example: **C:\Users\SSS IT\CubeGenerator** and write the following command on the command prompt:

1. mvn clean compile

Now, you will see a lot of execution on the command prompt. If you check your project directory, **target directory** is created that contains the class files.

## **Run the Maven Java Project**

To run the project, go to the project directory\target\classes,

for example: **C:\Users\SSS IT\CubeGenerator\target\classes** and write the following command on the command prompt:

1. java com.javatpoint.App

Now, you will see the output on the command prompt:

## **Output of the maven example**

1. Hello World!

## **How to build the maven project or how to package maven project?**

The **mvn package** command completes the build life cycle of the maven project such as:

1. validate
2. compile
3. test
4. package
5. integration-test
6. verify
7. install
8. deploy

Visit this link to know more about build life cycle <http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>

You need to execute the following command on the command prompt to package the maven project:

1. mvn **package**

Now you will see that **a jar file is created** inside the project/target directory.

You can also run the maven project by the jar file. To do so, go to the maven project directory, for example: **C:\Users\SSS IT\CubeGenerator** and execute the following command on the cmd:

1. java -classpath target\CubeGenerator-1.0-SNAPSHOT.jar;.; com.javatpoint.App

Now you will see the following output:

1. Hello World!

**Maven Web Application**

#### **Syntax**

The **syntax** to generate the project architecture is given below:

mvn archetype:generate -DgroupId=groupid -DartifactId=artifactid

-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=booleanValue

#### **Example**

The **example** to generate the project architecture is given below:

mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGeneratorWeb

-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

# **Maven Plugins**

The **maven plugins** are central part of maven framework, it is used to perform specific goal.

According to Apache Maven, there are 2 types of maven plugins.

1. Build Plugins
2. Reporting Plugins

#### **Build Plugins**

These plugins are executed at the time of build. These plugins should be declared inside the **<build>** element.

#### **Reporting Plugins**

These plugins are executed at the time of site generation. These plugins should be declared inside the **<reporting>** element.

## **Maven Core Plugins**

A list of maven core plugins are given below:

|  |  |
| --- | --- |
| **Plugin** | **Description** |
| clean | clean up after build. |
| compiler | compiles java source code. |
| deploy | deploys the artifact to the remote repository. |
| failsafe | runs the JUnit integration tests in an isolated classloader. |
| install | installs the built artifact into the local repository. |
| resources | copies the resources to the output directory for including in the JAR. |
| site | generates a site for the current project. |
| surefire | runs the JUnit unit tests in an isolated classloader. |
| verifier | verifies the existence of certain conditions. It is useful for integration tests. |

## **List of Maven Plugins**

To see the list of maven plugins, you may visit apache maven official website <http://repo.maven.apache.org/maven2/org/apache/maven/plugins/>. Maven plugins are also available outside the maven at **codehaus.org** and **code.google.com**.

# **Maven Interview Question**

A list of top frequently asked **maven interview questions** and answers are given below.

### 1) What is Maven?

Maven is a project management tool. It is based on POM (Project Object Model). [More details.](https://www.javatpoint.com/maven-tutorial)

### 2) What aspects are managed by Maven?

* Builds
* Documentation
* Reporting
* SCMs
* Releases
* Distribution

[More details.](https://www.javatpoint.com/maven-tutorial)

### 3) What are the advantages of Maven?

* No need to add jar file in each project
* Creates right directory structure
* Builds and deploys the project

[More details.](https://www.javatpoint.com/maven-tutorial)

### 4) What is the command to check the maven version?

Type the following command on console to know the maven version.

1. mvn -version

[More details.](https://www.javatpoint.com/how-to-install-maven)

### 5) What does the build tool?

* Generates source code (if the auto-generated code is used)
* Generates documentation from source code
* Compiles source code
* Packages compiled code into a JAR or ZIP file
* Installs the packaged code in the local repository, server repository, or central repository

### 6) What is the difference between Ant and Maven?

|  |  |
| --- | --- |
| **Ant** | **Maven** |
| It is **a toolbox**. | It is **a framework**. |
| It is **mainly a build tool**. | It is **mainly a project management tool**. |
| There is **no life cycle**. | There is a**life cycle**. |
| Ant **doesn't have formal conventions**. | Maven **has a convention** to place source code, compiled code etc. |
| Ant is **procedural**. | Maven is **declarative**. |
| The ant scripts are **not reusable**. | The Maven plugins are **reusable**. |

[More details.](https://www.javatpoint.com/difference-between-ant-and-maven)

### 7) Why is the use of the profile required in Maven?

For providing probability to projects, we use profiles.

### 8) What is the syntax for offline project creation?

The syntax for project creation is:

mvn o packg.

### 9) How is the propagation of plugins to child POMs stopped?

It can be done using the following syntax:

1. set**<inherited>** to false.

### 10) What is the use of the exclusion element?

The element is used to exclude dependencies.

### 11) Define SNAPSHOT in terms of maven.

The snapshot indicates the current development copy.

### 12) Define Archetype.

It is a Maven plugin which is designed for the creation of project structure.

### 13) Give the command for installation of the JAR file in a local repository.

mvn install

### 14) Mention the phases of cleaning lifecycle.

The lifecycle of cleaning consist of:

* pre-clean
* clean
* post-clean

### 15) What is the purpose of mvn clean command?

The command removes the target directory before the starting of a build process.

### 16) What is a MOJO?

A MOJO stands for Maven plain Old Java Object. Each MOJO is an executable goal in Maven, and a plugin is a distribution of one or more related MOJOs.

### 17) What is a repository?

A repository is a directory or place where all the jars and pom.xml file are stored. There are 3 types of a repository in Maven:

1. Local Repository
2. Central Repository
3. Remote Repository

[More details.](https://www.javatpoint.com/maven-repository)

### 18) What is a local repository?

Maven local repository is created by maven in your local system when you run any maven command. [More details.](https://www.javatpoint.com/maven-repository#local)

### 19) What is a central repository?

Maven community creates maven central repository on the web. [More details.](https://www.javatpoint.com/maven-repository#central)

### 20) What is a remote repository?

Maven remote repository is located on the web by different vendors. So you need to define the dependency in pom.xml file manually. It is important because most of the libraries are missing from the central repository

### 20) What is a remote repository?

Maven remote repository is located on the web by different vendors. So you need to define the dependency in pom.xml file manually. It is important because most of the libraries are missing from the central repository. [More details.](https://www.javatpoint.com/maven-repository#remote)

### 21) What is POM?

POM stands for Project Object Model. The pom.xml file contains information of project and project configuration. [More details.](https://www.javatpoint.com/maven-pom-xml)

### 22) What are the build phases in Maven?

1. validate
2. compile
3. test
4. package
5. integration-test
6. verify
7. install
8. deploy

### 23) What is the command to package maven project?

1. mvn -package

### 24) What is the fully qualified artifact name of maven project?

1. **<groupId>**:**<artifactId>**:**<version>**

### 25) What is an archetype?

Archetype is the maven plugin. It creates the project structure.

**……………………………………………………………………………………………**

**1) Explain what is Maven? How does it work?**

Maven is a project management tool. It provides the developer a complete build lifecycle framework. On executing Maven commands, it will look for POM file in Maven; it will run the command on the resources described in the POM.

**2) List out what are the aspects does Maven Manages?**

Maven handles following activities of a developer

• Build  
• Documentation  
• Reporting  
• Dependencies  
• SCMs  
• Releases  
• Distribution  
• Mailing list

**3) Mention the three build lifecycle of Maven?**

• Clean: Cleans up artifacts that are created by prior builds  
• Default (build): Used to create the application  
• Site: For the project generates site documentation

**4) Explain what is POM?**

In Maven, POM (Project Object Model) is the fundamental unit of work. It is an XML file which holds the information about the project and configuration details used to build a project by Maven.

**5) Explain what is Maven artifact?**

Usually an artifact is a JAR file which gets arrayed to a Maven repository. One or more artifacts a maven build produces such as compiled JAR and a sources JAR.

Each artifact includes a group ID, an artifact ID and a version string.

**6) Explain what is Maven Repository? What are their types?**

A Maven repository is a location where all the project jars, library jars, plugins or any other particular project related artifacts are stored and can be easily used by Maven.

Their types are local, central and remote

**7) Why Maven Plugins are used?**

Maven plugins are used to  
• Create a jar file  
• Create war file  
• Compile code files  
• Unit testing of code  
• Documenting projects  
• Reporting

**8) List out the dependency scope in Maven?**

The various dependency scope used in Maven are:

• Compile: It is the default scope, and it indicates what dependency is available in the classpath of the project  
• Provided: It indicates that the dependency is provided by JDK or web server or container at runtime  
• Runtime: This tells that the dependency is not needed for compilation but is required during execution  
• Test: It says dependency is available only for the test compilation and execution phases  
• System: It indicates you have to provide the system path  
• Import: This indicates that the identified or specified POM should be replaced with the dependencies in that POM’s section

**9) Mention how profiles are specified in Maven?**

Profiles are specified in Maven by using a subset of the elements existing in the POM itself.

**10) Explain how you can exclude dependency?**

By using the exclusion element, dependency can be excluded

**11) Mention the difference between Apache Ant and Maven?**

Apache Ant Maven  
• Ant is a toolbox – Maven is a framework  
• Ant does not have formal conventions like project directory structure – Maven has conventions  
• Ant is procedural; you have to tell to compile, copy and compress – Maven is declarative ( information on what to make & how to build)  
• Ant does not have lifecycle; you have to add sequence of tasks manually – Maven has a lifecycle  
• Ant scripts are not reusable – Maven plugins are reusable

**12) In Maven what are the two setting files called and what are their location?**

In Maven, the setting files are called settings.xml, and the two setting files are located at

• Maven installation directory: $M2\_Home/conf/settings.xml  
• User’s home directory: ${ user.home }/ .m2 / settings.xml

**13) List out what are the build phases in Maven?**

Build phases in Maven are

• Validate  
• Compile  
• Test  
• Package  
• Install  
• Deploy

**14) List out the build, source and test source directory for POM in Maven?**

• Build = Target  
• Source = src/main/java  
• Test = src/main/test

**15) Where do you find the class files when you compile a Maven project?**

You will find the class files ${basedir}/target/classes/.

**16) Explain what would the “jar: jar” goal do?**

jar: jar will not recompile sources; it will imply just create a JAR from the target/classes directory considering that everything else has been done

**17) List out what are the Maven’s order of inheritance?**

The maven’s order of inheritance is

• Parent Pom  
• Project Pom  
• Settings  
• CLI parameters

**18) For POM what are the minimum required elements?**

The minimum required elements for POM are project root, modelVersion, groupID, artifactID and version

**19) Explain how you can produce execution debug output or error messages?**

To produce execution debug output you could call Maven with X parameter or e parameter

**20) Explain how to run test classes in Maven?**

To run test classes in Maven, you need surefire plugin, check and configure your settings in setting.xml and pom.xml for a property named “test.”

………………………………………………………………………………………………….

**Q: What is Maven?**  
Maven is a build automation tool or a project management tool. With Maven we can import all libraries and can also create project structures. In Maven we have many inbuilt templates. These templates are called archetypes. A Maven is basically a tool used to compile our applications. It helps to develop and managing project structure or applications like deployment, clean, packaging, jar and many more features for the Java-based project.

In another word, it is a Java tool. If you want to create a sample project or skeleton project you can use Maven. It is an automated build tool. The Maven focused on simplicity that it generates intelligent starters and assumes intelligence defaults. It also covers build-oriented phases in Application Lifecycle Management i.e. testing, deployment, builds management, and release versioning.

**Q: How many project types available in Maven to choose from?**  
There is more than thousand Java project as there are templates, skeleton provided to you by Maven so that you do not have to remember a basic configuration detail or a basic setup of that particular type of project which Maven is going to give it to you. It includes examples like basic Java project, Spring Project, Spring MVC, Spring Web Flow, and Spring Boot.

**Q: Why should one use Maven?**  
>> It helps to setup project very quickly and it avoids complicated build files like build.xml. Maven required files like POM.xml; it serves the purpose for Maven only. POM.xml is a collection of dependencies of your Java Project which one can specify to Maven and then Maven will download all of them from the internet and then store it to some repository i.e. local repository, central repository, and remote repository.

>> It helps to not bundle all the jars in your package i.e. in your War file or Ear file because all of them are going store in the repository and wherever you install this application that repository will be used for any dependencies look up. So, your Jar file, War file or Ear file or your bundle deployment will be very light.

**Q: Maven advantages over Ant?**  
\*\* Maven uses Convention whereas ant uses Configuration. In Maven, convention means a standard layout suggested by Maven. If you want a skeleton project of sample Java application, so Maven is going to give you one set of folders and configurations for that Java applications whereas in the case of Ant you need to have lots of configurations.  
\*\* Maven supports project modularization.  
\*\* Maven also supports dependency management and migration.

**Q: What Maven creates for you?**  
1. Directory Name  
2. Purpose  
3. Project home  
4. Contains the POM.xml and all subdirectories.  
5. Src/main/Java  
6. Contains the deliverable Java source code for the project.  
7. Src/main/resources  
8. Contain the deliverable resources for the project.  
9. Src/test/Java  
10. Contains the testing Java source code.  
11. Src/test/resources  
12. Contains resources necessary for testing.

**Q: For whom this Maven course is for?**  
>> Maven can use for the Intermediate Java developers.  
>> It works for a large project or portfolio managers.  
>> It helps to work for heavy users of libraries.

**Q: What are the Maven Phases?**  
1. The first phase is the Validate Phases. It checks whether everything is in order i.e. the configuration is running properly, the code is placed in a proper way. This kind of validation check is done in the validate phase.  
2. The next comes the compiler phases. It compiles everything together and stores it.  
3. The third comes to the test phases. Here we run the test cases which are specified for the code.  
4. The fourth phase is the package phases. Package file ends up in Jar file or War file or Ear file depending on what we have specified on the POM.xml.  
5. The fifth phase is the install phase. It installs to your local Maven repository.  
6. The sixth phase is called the deploy phase.

**Q: What are the tenets of Maven?**  
1. Project oriented.  
2. Dependency Management.  
3. Reuse through centralized repositories.  
4. Convention over Configuration.  
5. Extensible through plugins.

**Q: What is the Projected Oriented Build in Maven?**  
Maven tools are build on the model as POM i.e. Project Object Model.  
Supports single-inheritance tree like Java.  
Defined as XML in pom.xml  
Every pom extends the super POM

**Q: What are POM Files in Maven?**  
All your code and resources are placed in the src directory.  
>> The main/Java directory holds your project code.  
>> Compiled code is placed in the largest directory.  
>> The test/Java directory holds your JUnit test code.

**Q: What is Maven Archetypes? And what are the Project Types?**  
Archetypes are templates to create a variety of Java project structures, including web applications specific to a container such as Wildfly. In other words, it is a tool that creates the stuff you build the project on top of.

**Project Types:**  
1. War  
2. Jar  
3. Ear  
4. Wildfly  
5. JMS  
6. Android

**Q: What are Maven’s main objectives?**  
\*\* It helps to make project build easy to work for the users.  
\*\* It easily helps in to migrate from one feature to another or one folder to another.  
\*\* It helps in the new development with proper series of guidelines.  
\*\* It creates a flexible working system for the users. So that you can work uniformly and orderly.

**Q: What is Maven Repository?**  
In Maven a repository is used as a storage folder or a directory to store your projects, your files such as Jar, War or Ear files that can be later used by the Maven application or tool. It works as a whole library of the files that is easily accessible and can be easily located in your system without any trouble and then can be used by Maven.

**Q: How many repositories are there in Maven?**  
There are three types of repository present in Maven. This includes Local Repository, Central Repository, and Remote Repository.  
**Local Repository-** This local repository is located on your local system and it works when you run a maven command. Maven local repository command is %USER\_HOME%/.m2 directory.  
**Central Repository-** Installation from the repository is performed on creating a project from archetype or resolving the dependency.  
**Remote Repository-** This repository is located on the web. It is just a network accessible location that Maven downloads dependencies from. All the artifacts that remote repository contains are open source.

**Q: How to install Maven?**  
Make sure JDK is installed, and ‘JAVA\_HOME’ variable is added as Windows environment variable.  
Add both M2\_HOME and MAVEN\_HOME variable in the Windows environment, and point it to your Maven folder.

**Q: What is the Maven Lifecycle?**  
Lifestyle executed in term of phases:  
1. Maven Steps through phases.  
2. Execution defined in terms of plugin goals.  
3. Execution associated with phases.  
4. Lifecycle completes when all phase executes successfully.

**Q: What is the system requirement for Maven?**  
Maven does not require any high configuration to use. It requires only very minimal and simple system requirements for the users:  
1. Java Deployment Kit  
2. Installed  
3. Configured (JAVA\_HOME)  
4. Internet Connection  
5. For interacting with the repository.  
6. Downloading dependencies.

Advanced Interview Questions on Maven

**Q: What is the method and installing process of Maven?  
Download Maven from:**  
https://maven.apache.org./download.html  
Choose the .zip format.

**Extract Maven to:**  
/usr/local/maven – Unix / Linux  
C:Program Filesmaven – Windows

**Q: What is Key Maven Command?**  
Maven as an application has a limited number of commands  
“Command” is a way of invoking the maven lifecycle.  
Other “Commands” are defined through plugins.

**Q: What is Maven Plugin?  
Core Maven functionality is simplistic:**  
Really just a plugin execution framework.  
Knows predefined lifecycle and how to execute plugins.  
Plugins are dynamically downloaded and installed.

**Plugins encapsulate build related functionality:**  
Define by name.  
Contain a set of goals.  
Plugins are invoked using the syntax:  
%>mvn plugin\_name:goal

**Q: What are the archetype goals?**  
Four goals associated with archetype plugin:  
**Create -** creates using a quick-start template.  
**Generate –** provide a menu of templates.  
**Create-from-project –** creates an archetype from an existing project.  
**Crawl –** searches the repository for archetype and updates catalog.

**Q: What are Maven build plugins?  
Antlr Plugin –** Generates parsers that you can use in your code from a very concise domain-specific language.  
**QueryDSL Plugin –** Interrogates your database and creates data access objects that you can use to write SQL- like queries in your Java code.  
**Test Report Plugins –** Make cool report websites that show you how well your unit tests are covering your code.  
**Shade Plugin –** Allow you to bundle all of your classes into an uber-Jar that is runnable. This way you can deploy just one artifact. Also, can change packages of your dependencies to work around conflicts.

**Q: What is the dependency scope in Maven?  
Compile**– The library is available while compiling and running, main code and test code.  
**Provided** – The library is available at compile time but will not be packaged with your code at run-time. Typically this is used when running on an app-server that will provide its chosen version of the library to all running applications.  
**Runtime** – The library is not around for compilation but is around at run-time.   
**Test** – Only include the library when compiling/running unit tests.  
**System** – Refer to a library on this computer by filename; useful for libraries built into system.

**Q: What is parent POMs?**  
1. Basically, these are parent projects without code.  
2. Used by companies to define the set of libraries/versions, plugins they want their teams using.  
3. Can have dependencies, build plugins, variables definitions, and even their own parent POM, forming a chain.  
4. A great example is Spring Boot. You can extract it to create production-grade web services crazily fast.

**Q: What is Maven Artifact?**  
Maven Artifact consists of files like Jar file or War file that result in the expansion of the specific file in the Maven repository. The Jar file can use as an artifact in Maven. The Maven Artifact determined by a group ID name to run the Maven Artifact in the Maven. They can contain files like Ear, Jar, and War or Zip file as well. Maven Artifact is used for specifying applications to locate a name or package. It is usually stored in your system repository.

**Q: What is the sequence in which Maven searches for dependency libraries?**  
You can locate dependency in the local repository system of your software. Sometimes, it is difficult to locate or identify in the local repository. So, I can find or look in the central repository system and if it shows the dependency missing then one can look in remote repository to find the dependency. If it still shows the same thing repeatedly then the system will show error in finding the dependencies. And if the dependencies are found in the local repository then it will be automatically downloaded in the central repository for future use.

**Q: What are the things you need to define for each external dependency?**  
External Dependency plays an important part in the Maven software. It is an internal part of the system without which dependency cannot be located in a system. To specify the external dependency we need:  
1. It requires a group ID duplicate to the library name.  
2. It requires an artifact ID duplicate to the library name.  
3. Mentioning of dependency scope in the system  
4. Have to mention the system route corresponding to the project position.

**Q: What are the steps involved in project deployment?**  
There are several steps to follow while implying project deployment in Maven. These steps include:  
1. Go through all the projects and analyze the code that is working in the background in progress in the source code repository and identifies it.  
2. In order to get the project development, one need to download the whole source code from the Social Venture Network.  
3. Construct or develop the application in the system  
4. It needs to be saving as a War or Jar file system.  
5. Get the specified file from the location path and move that specific file to create a site.  
6. The application that is created in the system needs to be updated with the latest version with date and version number.

**Q: What are the aspects Maven manages?**  
1. Documentation  
2. SCMs  
3. Distribution  
4. Builds  
5. Reporting  
6. Releases  
7. Mailing list  
8. Dependencies

**Q: What is a goal in Maven terminology?**  
The goals mentioned here in the Maven suggests the managing and building process requires creating a project. There is no limitation to follow the goals in Maven; it can build as many phases as it wants with zero boundations. You directly achieve your goal without any kind of outside intervention.

…………………………………………………………………………………………………………